4 / 02 / 2020

MACHINE LEARNING UDEMY

LINEAR REGRESSION SCIKIT LEARN

- SUPERVISED LEARNING
- REGRESSION TOWARDS A MEAN
- DATA → LINEAR REGRESSION → MODEL

SUM of SQUARED ERRORS
SUM of ABSOLUTE ERRORS
SUM of SQUARES of the RESIDUALS

X = FEATURES (NUMBER of ROOMS)
Y = WHAT YOU WANT TO PREDICT (PRICE)

```
X_train, X_test, y_train, y_test
    = train_test_split(x, y, test_size
        = 0.4, random_state=101)
```

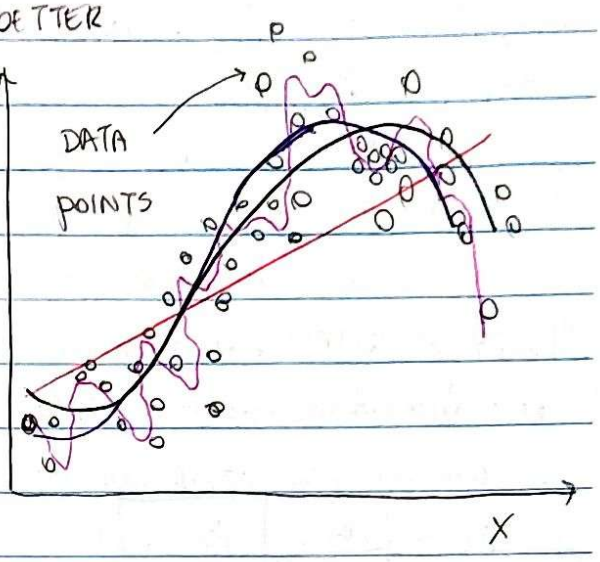
TRAIN TEST SPLIT : SPLITS DATA IN TWO, ONE PART to TRAIN the MODEL AND the other to TEST IT. TEST SIZE is the PERCENTAGE of TOTAL DATA that WILL BECOME the TEST DATA. RANDOM STATE is how SCIKIT LEARN SPLITS DATA.

```
lm = LinearRegression()
lm.fit(X_train, Y_train)
lm.coef_
```

RECOMMENDED BOOK : INTRODUCTION to STATISTICAL LEARNING BY GARRETH JAMES, CHAPTERS 1 AND 2 (ISL BOOK)

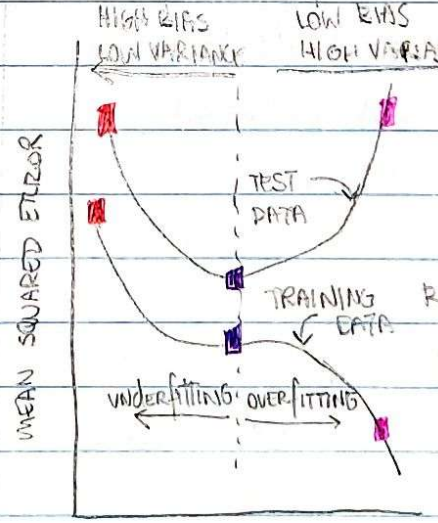
BIAS VARIANCE TRADE OFF

INCREASING the COMPLEXITY of A MODEL does NOT NECESSARILY MEAN the MODEL IS BETTER



LINEAR REAL CURVE
QUADRATIC
SPLINE

SEE HOW, IN this CASE, the MOST COMPLEX MODEL ENCAPTURES A LOT of DATA POINTS, BUT IS FARTHER from the REAL CURVE



for the TRAINING DATA, USING SPLINE REDUCES the ERRORS, BUT IT WILL NOT REPRESENT DATA BETTER

4 / 03 / 20

MACHINE LEARNING udemy

LOGISTIC REGRESSION

INTRODUCTION TO STATISTICAL LEARNING, SECTIONS 4-4.3

- CLASSIFICATION; DISCRETE CATEGORIES
- SIGMOID FUNCTION

$$\phi(t) = 1/(1+e^{-t})$$

$$p = \frac{1}{1+\exp-(a+bx)}$$

→ VALUES RANGE FROM 0 to 1

- CONFUSION MATRIX → MODEL EVALUAT.
  → DESCRIBES THE PERFORMANCE OF THE CLASSIFICATION MODEL

n=165	PREDICTED FALSE	PREDICTED TRUE
ACTUAL FALSE	TN = 50	FP = 10
ACTUAL TRUE	FN = 5	TP = 100

TP = TRUE POSITIVE
TN = TRUE NEGATIVE
FP = FALSE POSITIVE = TYPE 1 ERROR
FN = FALSE NEGATIVE = TYPE 2 ERROR

- ACCURACY: how often is correct?

$\frac{TP+TN}{TOTAL}$ = ACCURACY = 91%

- ERROR RATE (MISCLASSIFICATION RATE)
  how often is it wrong? $\frac{FP+FN}{TOTAL}$ = 9%

SCIKIT LEARN

sklearn.metrics.confusion_matrix
sklearn.metrics.classification_report
↳ PRECISION, RECALL, F1 SCORE, SUPPORT

PRECISION: ABILITY of the CLASSIFIER NOT TO LABEL AS positive A SAMPLE THAT IS NEGATIVE: $\frac{TP}{TP+FP}$

RECALL: ABILITY of the CLASSIFIER to find ALL positives: $\frac{TP}{TP+FN}$

F1 SCORE: WEIGHTED HARMONIC MEAN of PRECISION AND RECALL
0 → WORST CASE, 1 → BEST

SUPPORT: NUMBER of OCCURENCES of EACH CLASS IN y-true (CORRECT VALUES)

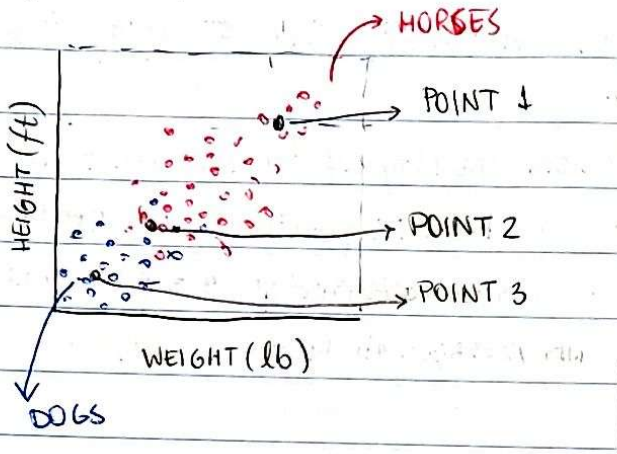
4 / 04 / 20

MACHINE LEARNING UDEMY

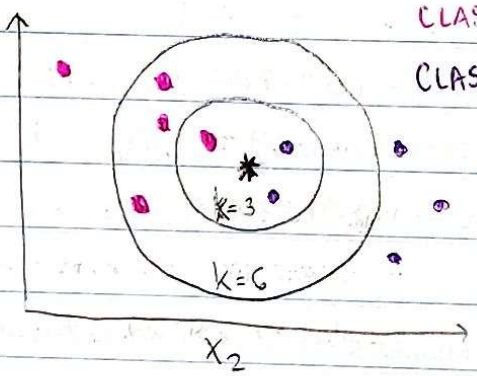
K NEAREST NEIGHBORS (KNN)

- INTRODUCTION TO STATISTICAL LEARNING CHA
- CLASSIFICATION ALGORITHM BASED ON THE NEIGHBORING POINTS CALCULATING THE DISTANCE FROM x TO ALL POINTS

EXAMPLE: NEW POINTS: DOG OR HORSE?



K NUMBER AFFECTS THE CLASS A NEW POINT GOES INTO:



k=3 ∴ NEW POINT IS FROM CLASS B

k=6 ∴ NEW POINT IS FROM CLASS A

↓

COMPARES HOW MANY NEIGHBORS THERE ARE

THE GREATER THE K NUMBER, MORE BIAS, BETTER CUT OFF BUT ALSO MORE MISLABELED POINTS

1.

SIMPLE MODEL WITH FEW PARAMETERS, BUT ITIS NOT RECOMMENDED FOR HIGH DIMENSIONAL DATA AND LARGE DATA SETS AND CATEGORICAL FEATURES

→ SCALE DATA WITH sklearn.preprocessing. STANDARDSCALER TO STANDARDIZE

→ find OPTIMAL K NUMBER by COMPARING ERROR RATES (y_test MINUS predictions)

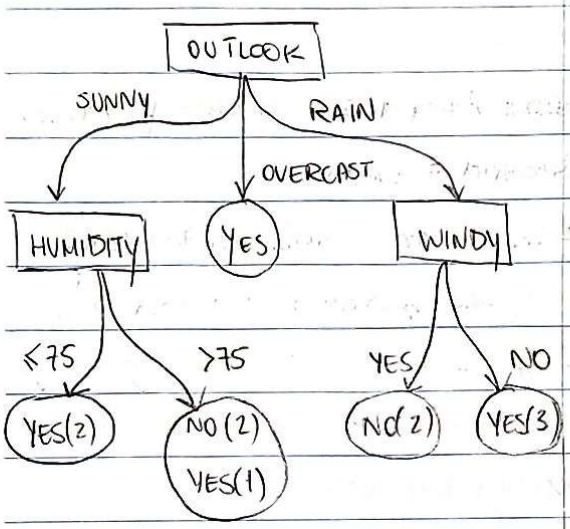
4/07/20

MACHINE LEARNING UDEMY

DECISION TREES AND RANDOM FORESTS

- INTRODUCTION TO STATS LEARN. CH8

DECISION TREES: CLASSIFICATION & REGRESSION; NUMERICAL & CATEGORICAL



→ NODES (OUTLOOK, HUMIDITY, WINDY)
→ EDGES (NODE OUTCOME)
→ ROOT (OUTLOOK)
→ LEAVES ("RESULT"; YES/NO)
→ ENTROPY & INFORMATION GAIN

DISADVANTAGES
→ MAKES MOST OPTIMAL DECISION ON EACH STEP, BUT NOT THE GLOBAL OPTIMUM
→ PRONE TO OVERFITTING

RANDOM FORESTS: COLLECTION OF DECISION TREES
→ MINIMIZATION OF ERRORS BY BIAS AND/OR VARIANCE
→ LIMIT OVERFITTING

DIFFERENCES:

DECISION TREE	RANDOM FOREST
→ BUILT ON ENTIRE DATASET USING ALL FEATURES	→ RANDOMLY SELECTS FEATURES AND OBSERVATIONS

THE RANDOM FOREST BUILDS MULTIPLE DECISION TREES WITH A NUMBER OF FEATURES AND OBSERVATIONS AND AVERAGES THE RESULTS

DECISION TREE: sklearn.tree.DecisionTreeClassifier
RANDOM FOREST: sklearn.ensemble.RandomForestClassifier

rfc = RandomForestClassifier(PARAMETERS)
↳ n_estimators, max_features

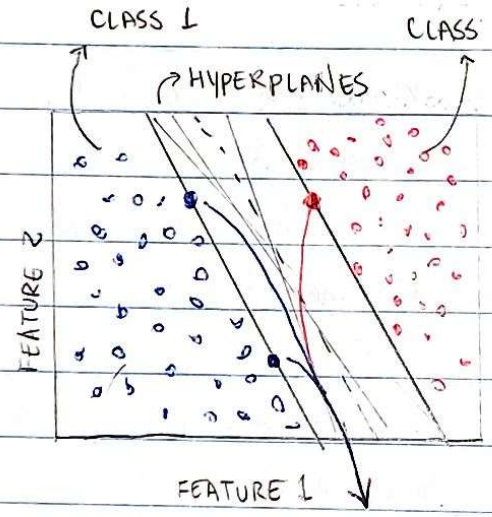
- n_estimators: number of trees to be used in the forest
- max_features: max number of FEATURES to be CONSIDERED:
  - → REGRESSION: USE n_FEATURES
  - → CLASSIFICATION: USE $\sqrt{n\_features}$

4 / 12 / 20

MACHINE LEARNING UDEMY

SUPPORT VECTOR MACHINES (SVM)

- ISL CHAPTER 9
- SUPERVISED LEARNING to RECOGNIZE PATTERNS FOR CLASSIFICATIONS AND REGRESSION
- NON PROBABILISTIC BINARY LINEAR CLASSIFIER → CLASSIFICATION
- MANY HYPERPLANES CAN EXIST BETWEEN CLASSES
- GOAL: CHOOSE A HYPERPLANE that MAXIMIZES the MARGIN BETWEEN CLASSES



- the VECTOR POINTS THAT TOUCH THE MARGIN LINES ARE THE SUPPORT VECTORS
- KERNEL TRICK → NON LINEARLY SEPARABLE DATA
- GRID SEARCH: GRID of PARAMETERS USED to TRY COMBINATIONS to find GOOD SVM PARAMETERS

```
from sklearn.model_selection import GridSearchCV

param_grid =
  {"C": [0.1, 1, 10, 100, 1000],
   "gamma": [1, 0.1, 0.01, 0.001, 0.0001]}

grid = GridSearchCV(estimator, param_grid,
                    verbose, refit)
```

→ estimator = SVC()
needs to provide A SCORE FUNCTION

→ param_grid = param_grid
DICTIONARY WITH PARAMETER NAMES (i.e. C and GAMMA) AND VALUES
- (LIST of SETTINGS to BE TESTED)

C: CONTROLS the COST of MISCLASSIFICATION ON THE TRAINING DATA
↑C ∴ LOW BIAS, HIGH VARIANCE, HIGH PENALIZATION

GAMMA: FREE PARAMETER of the GAUSSIAN RADIAL BASIS FUNCTION (kernel='rbf')
↑GAMMA ∴ HIGH BIAS, LOW VARIANCE; SUPPORT VECTOR WON'T HAVE A WIDESPREAD INFLUENCE

→ verbose = 3 (default zero)
the higher the VERBOSE, the MORE MESSAGES

→ refit = True
REFIT AN ESTIMATOR USING the BEST found PARAMETERS ON THE WHOLE DATASET

4 / 13 / 2020

MACHINE LEARNING UDEMY

K MEANS CLUSTERING

- ISL CHAPTER 10
- UNSUPERVISED LEARNING
- DIVIDE DATA INTO DISTINCT GROUPS

ALGORITHM

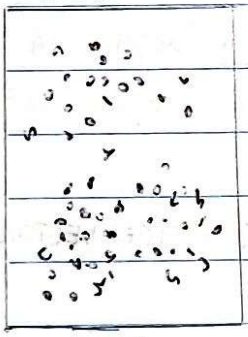
→ CHOOSE A NUMBER OF CLUSTERS K

→ RANDOMLY ASSIGN EACH POINT TO A CLUSTER

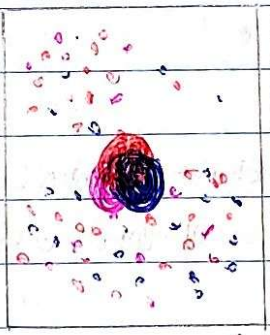
→ ITERATE OVER THE COMPUTED CENTROIDS UNTIL SSE (SUM OF SQUARED ERRORS) DOES NOT PLUNGE WHEN COMPARED TO THE K NUMBER

EXAMPLE: TAKE UNLABELED DATA AND ASSIGN EACH POINT TO A RANDOM CLUSTER. COMPUTE THE CENTROID

UNLABELED DATA

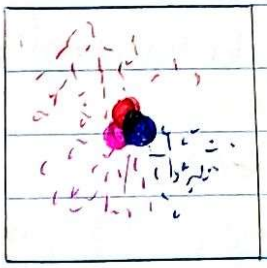


STEP 1

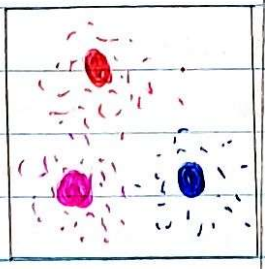


ASSIGN THE DATA POINTS TO THE CLUSTER FOR WHICH THE CENTROID IS CLOSEST

ITERATIONS

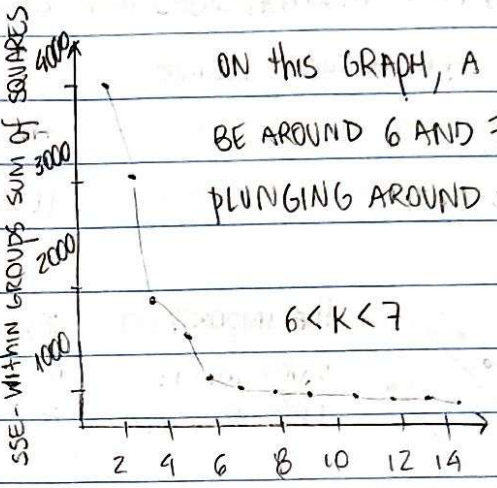


RESULT



CHOOSING K VALUE

- SSE: $\sum (\text{POINT} - \text{CENTROID})^2$
- ↑ NUMBER of CLUSTERS ∴ ↓ SIZE of CLUSTERS
- CHOOSE A K AT WHICH the SSE DECREASES ABRUPTLY (ELBOW EFFECT)



ON this GRAPH, A GOOD K WOULD BE AROUND 6 AND 7, AS IT STOPS PLUNGING AROUND these VALUES

4 13 20

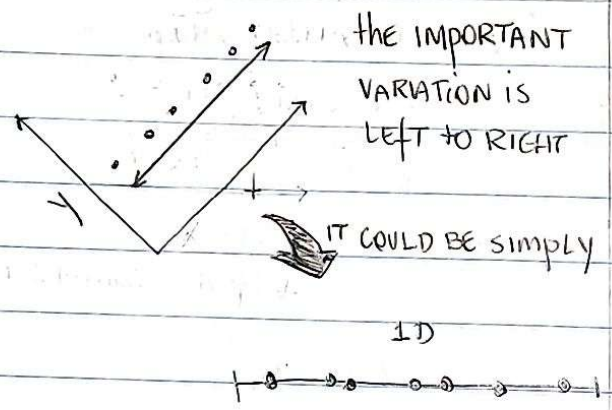
MACHINE LEARNING UDEMY

PRINCIPAL COMPONENT ANALYSIS

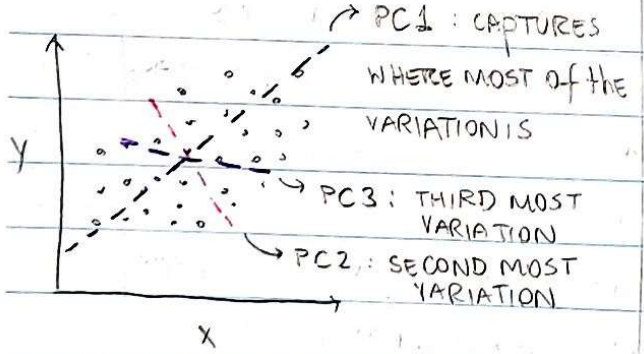
- ISL CH 10.2
- UNSUPERVISED LEARNING
- COMPRESSION OF MANY DATA DIMENSIONS INTO A FEW THAT ARE THE MOST RELEVANT

MAIN IDEA: FLATTEN n-DIMENSIONS INTO A FEW WITHOUT LOSING VALUE OF DATA

EXAMPLE: 2D DATA:



EXAMPLE: take the GREATEST VARIATIONS ON the LINES THAT REPRESENT DATA



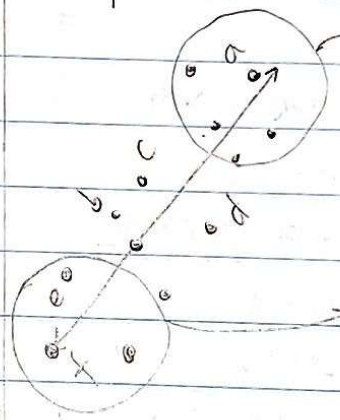
- the PRINCIPAL COMPONENTS ARE A combination of ALL OR SOME of the FEATURES

EXAMPLE: sklearn BREAST CANCER DATASET
→ 30 FEATURES (RADIUS, AREA, SYMMETRY...)

CANCER DATA CAN BE TRANSFORMED WITH PCA to HAVE ONLY two PRINCIPAL COMPONENTS. PC1 AND PC2 ARE COMBINATIONS of WEIGHTS of EACH of the 30 FEATURES

the CLOSER the PC VALUE IS to ZERO, the LESS INFLUENCE (WEIGHT) THAT POINT HAS

EXAMPLE:



POINT	INFLUENCE	VALUE
a	HIGH	10
b	LOW	0.5
c	LOW	0.2
d	LOW	-0.2
e	HIGH	13
f	HIGH	-14

→ WEIGHT = LOADING
→ ARRAY of LOADING for PC = EIGENVECTOR

STEPS:

→ STANDARDIZE DATA
sklearn.preprocessing.StandardScaler
scaler.fit(df) and scaler.transform(df)
→ PCA
sklearn.decomposition.PCA
pca.fit(scaledData) and pca.transform

4 / 19 / 2020

MACHINE LEARNING UDEMY

RECOMMENDER SYSTEMS

→ RECOMMENDER SYSTEMS by JANNACH AND ZANKER

→ COLLABORATIVE FILTERING (CF)
RECOMMENDATIONS BASED ON USERS RATING; "AUTO" FEATURE LEARNING

→ CONTENT-BASED
RECOMMENDATIONS BASED ON ATTRIBUTES OR SIMILARITIES of ITEMS

NATURAL LANGUAGE PROCESSING

• COMPILE AND FEATURIZE DOCUMENTS

EXAMPLE: DOCS "BLUE HOUSE" AND "RED HOUSE"

→ FEATURIZE BY WORD COUNT
(red, blue, house)
"BLUE HOUSE" → (0, 1, 1)
"RED HOUSE" → (1, 0, 1)

• BAG of WORDS: A DOCUMENT REPRESENTED AS A VECTOR

WORD/DOCUMENT SIMILARITY

$$\text{SIMILARITY}(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

• BAG of WORDS CAN BE IMPROVED BY ADJUSTING WORD COUNTS BASED ON THEIR FREQUENCY IN CORPUS (GROUP of ALL DOCUMENTS)

• TF-IDF: TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY
↳ COUNT HOW MANY TIMES A WORD OCCUR

→ TERM FREQUENCY (TF): IMPORTANCE of term t IN document d ∴ $TF = f(d,t)$
↳ WEIGHTS COUNTS SO FREQUENT TOKENS GET LOW WEIGHT

→ INVERSE DOCUMENT FREQUENCY (IDF): IMPORTANCE of term t IN CORPUS (GROUP of ALL DOCS)
∴ $IDF(t) = \log(D/t)$
t: NUMBER of DOCUMENTS WITH TERM
D: total NUMBER of DOCUMENTS

→ TF-IDF (W): term x WITHIN document y

$$W_{x,y} = t_{x,y} \cdot \log\left(\frac{N}{df_x}\right)$$

$t_{x,y}$: FREQUENCY of x IN y
df_x: NUMBER of DOCS CONTAINING x
N: total of docs

this CALCULATION CONSIDERS how IMPORTANT A WORD IS BESIDES ITS COUNT

• STOP WORDS: FREQUENT WORDS USED IN A LANGUAGE; UNIMPORTANT TO NLP

• TOKENIZATION: CONVERSION of PURE TEXT INTO TOKENS (the IMPORTANT PART of the TEXT) BY REMOVING STOP WORDS AND PUNCTUATION, i.e

• STEMMING AND LEMMATIZATION
the boy's cars are different colors
the boy car be differ color

4 / 24 / 2020

MACHINE LEARNING UDEMY

NEURAL NETS AND DEEP LEARNING

TOPICS:

- PERCEPTRON MODEL
- ACTIVATION FUNCTIONS
- COST FUNCTIONS
- FEED FORWARD NETWORKS
- BACK PROPAGATION

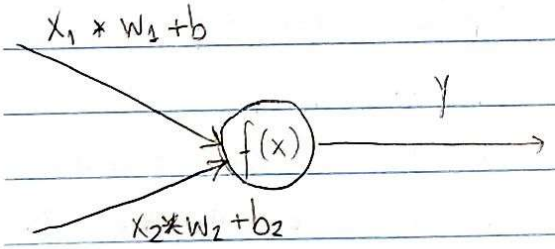
CODING TOPICS:

- TENSORFLOW 2.0 (KERAS)
- ARTIFICIAL NEURAL NETWORKS
  - ↳ REGRESSION
  - ↳ CLASSIFICATION

PERCEPTRON MODEL

IDEA:

INPUT → FUNCTION → OUTPUT



$$\hat{y} = \sum_{i=1}^{n} X_i W_i + b_i$$

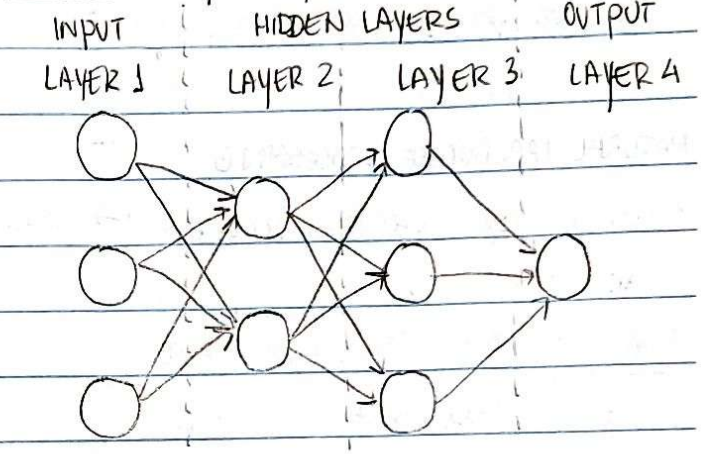
where: X_i = INPUT, W_i = WEIGHT

b_i = BIAS, $\hat{y}$ = OUTPUT

WEIGHT (w): ADJUSTS INPUT X_i STRENGTH

BIAS (b): OFFSET TO MAKE $X_i * W_i$ REACH A CERTAIN THRESHOLD BEFORE HAVING AN EFFECT; SHIFTS THE ACTIVATION FUNCTION.

MULTILAYER PERCEPTRON MODEL



- OUTPUT OF A LAYER IS THE INPUT OF ANOTHER
  - ↳ INTERACTION AND RELATIONSHIPS BETWEEN FEATURES
- DEEP NEURAL NETWORK if THERE ARE 2 OR MORE HIDDEN LAYERS
- UNIVERSAL APPROXIMATION THEOREM: ANY CONVEX CONTINUOUS FUNCTION CAN BE APPROXIMATED BY A FEED-FORWARD NETWORK WITH A SINGLE HIDDEN LAYER CONTAINING A FINITE NUMBER OF NEURONS

4 / 24 / 2020

MACHINE LEARNING UDEMY - NEURAL NETS AND DEEP LEARNING

ACTIVATION FUNCTIONS

- they SET BOUNDARIES TO OUTPUT VALUES from the NEURON
- $z = x * w + b$

x → INPUT

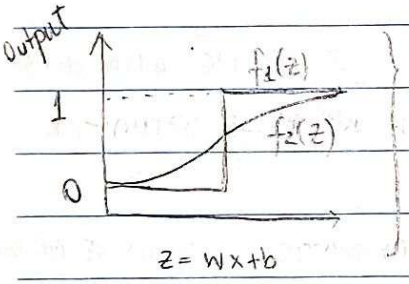
w → WEIGHT (INPUT STRENGTH)

b → BIAS (offset)

i.e. if $b = -10$: the effects of $x * w$ WON'T START to OVERCOME the BIAS UNTIL their PRODUCT IS GREATER (than) 10

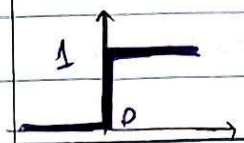
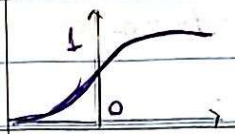
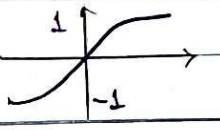
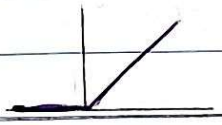
ACTIVATION FUNCTIONS LIMITS z

EXAMPLE: CLASSIFICATION problem (OUTPUT IS EITHER 0 or 1)



Apply the ACTIVATION FUNCTION TO GET z

TABLE of COMMON ACTIVATION FUNCTIONS

NAME	PLOT	EQUATION
BINARY STEP		$\sigma(z) = \begin{cases} 0, z<0 \\ 1, z \geq 0 \end{cases}$ {0,1}
SIGMOID		$\sigma(z) = \frac{1}{1+\exp(-z)}$ (0,1)
tanh		$\sigma(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$ (-1,1)
RECTIFIED LINEAR UNIT		$\sigma(z) = \begin{cases} 0, z \leq 0 \\ z, z > 0 \end{cases}$ [0,∞)
SOFTMAX	PROBABILITY DISTRIBUTION OF EVENTS OVER K EVENTS. PROBABILITY OF TARGET CLASS OVER ALL CLASSES	$\sigma(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}$ $i = 1 : k$ (0,1)

MUTUALLY EXCLUSIVE CLASS : i.e. SOFTMAX

		RED	GREEN	BLUE
DATA POINT 1	RED	1	0	0
DATA POINT 2	GREEN	0	1	0
⋮		...	...	...
DATA POINT N	BLUE	0	0	1

MULTICLASS CLASSIFICATION

- NON EXCLUSIVE CLASS
  - ↳ A data POINT CAN HAVE MULTIPLE CLASSES / CATEGORIES (i.e. PHOTO TAGS)
- MUTUALLY EXCLUSIVE CLASSES
  - ↳ ONLY ONE CLASS PER DATA POINT (i.e. PHOTO IS B&W OR COLOR)

NON-EXCLUSIVE CLASSES : i.e. SIGMOID

		A	B	C	
DATA 1	A, B	1	1	0	CLASS 1 : 0.8
DATA 2	A	1	0	0	CLASS 2 : 0.3
DATA 3	C, B	0	1	1	CLASS 3 : 0.5
⋮		...	...	...	⋮
DATA N	B	0	1	0	

4 / 24 / 2020

MACHINE LEARNING UDEMY - NEURAL NETS AND DEEP LEARNING

COST FUNCTIONS & GRADIENT DESCENT

- COST = $C(W, B, S^r, E^r)$

where C = COST, W = WEIGHT, B = BIAS

S^r = INPUT of SINGLE TRAINING SAMPLE

E^r = DESIRED OUTPUT of TRAINING SAMPLE

- FORMULAS : QUADRATIC COST FUNCTION

$$C = \frac{1}{2n} \sum_x \| y(x) - a^L(x) \|^2$$

$$a = \sigma(z)$$

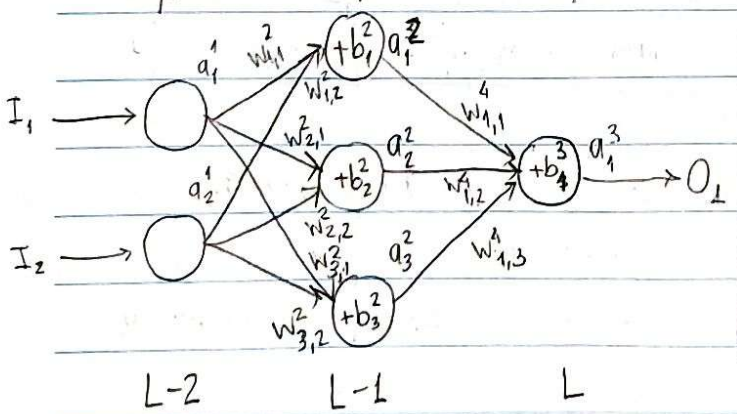
$$z = wx + b$$

where: $y(x)$: TRUE VALUE

$a^L(x)$: NEURON'S PREDICTION OR ACTIVATION OUTPUT FOR LAYER L (LAST LAYER)

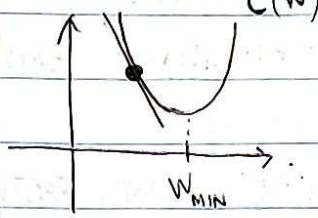
$\sigma(z)$: ACTIVATION FUNCTION

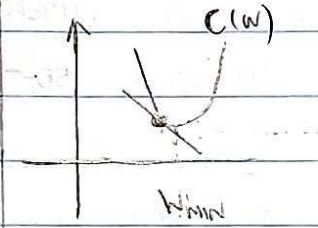
n: NUMBER OF POINTS



GOAL : MINIMIZE the COST (IT'S the DIFFERENCE BETWEEN the REAL OUTPUT AND the PREDICTION) ∴ DERIVATIVE AND SOLVE FOR ZERO ∴ GRADIENT DESCENT

SIMPLIFIED EXAMPLE: CALCULATE SLOPE AND MOVE IT DOWNWARD UNTIL CONVERGE TO ZERO





STEP SIZE = LEARNING RATE. IF IT ISN'T CONSTANT, THEN IT'LL BE AN ADAPTIVE GRADIENT DESCENT (∇C)

ADAM : METHOD FOR STOCHASTIC OPTIMIZATION. UPTADE of RMSPROP OPTIMIZER

STOCHASTIC APPROXIMATION : ITERATIVE METHODS FOR ROOT-FINDING OR OPTIMIZATION

DERIVATIVE → VECTORS

GRADIENTS → TENSORS

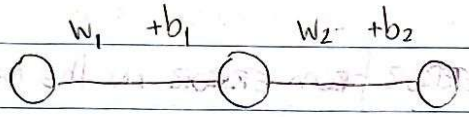
FOR CLASSIFICATION PROBLEMS: CROSS ENTROPY (MEASURE OF DIFFERENCE BETWEEN TWO PROBABILITY DISTRIBUTIONS FOR EVENTS)

MACHINE LEARNING UDEMY - NEURAL NETS AND deep LEARNING

BACKPROPAGATION

- EVALUATES THE EXPRESSION FOR the DERIVATIVE of the COST FUNCTION AS A PRODUCT of DERIVATIVES BETWEEN EACH LAYER BACKWARDS WITH THE GRADIENT of the WEIGHTS BETWEEN EACH LAYER BEING A SIMPLE MODIFICATION of THE PARTIAL PRODUCTS (the BACKWARDS PROPAGATED ERROR)

EXAMPLE: $C(w_1, b_1, w_2, b_2)$



$z = wx + b$

ACTIVATION function $a = \sigma(z)$

$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma(z^L)$$

$$C_0 = (a^L - y)^2$$

BACKPROPAGATION

HOW SENSITIVE IS the COST FUNCTION CHANGE WITH RESPECT TO WEIGHT/BIAS?

$$\frac{\partial C_0}{\partial w^L} \quad \text{and} \quad \frac{\partial C_0}{\partial b^L}$$

GOAL: MINIMIZE the OUTPUT of the ERROR VECTOR ON LAYER L BY ADJUSTING WEIGHT AND BIAS GOING BACKTHROUGH the NETWORK USING the GRADIENT

ERROR VECTOR: $\delta^L = \nabla_a C \odot \sigma^l(z^L)$

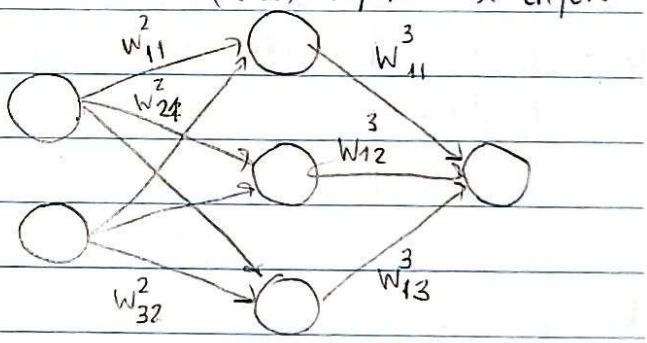
where L = LAST LAYER

l = CURRENT MIDDLE LAYER

$\odot$ = HADAMARD PRODUCT

$\nabla_a C = (a^L - y)$

BACK TO the NEURAL NETWORK EXAMPLE:



NOTATION: w^l_{jk}, b^l_j, a^l_j

j: NEURON IN the l^{th} LAYER

k: NEURON IN the $(l-1)^{th}$ LAYER

l: LAYER

w^l_{jk}: WEIGHT FROM k to j LAYER

b^l_j: BIAS of NEURON j IN LAYER l

a^l_j: ACTIVATION of NEURON j IN LAYER l

4 / 25 / 2020

MACHINE LEARNING UDEMY - NEURAL NETS AND DEEP LEARNING

$$a^l = \sigma\left(\sum_k w^l_{jk} \cdot a^{l-1}_k + b^l_j\right)$$

ACTIVATION ON LAYER l IS BASED ON THE ACTIVATION OF THE PREVIOUS $(l-1)$ LAYER

MATRIX FORM:

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

where:

$$w^l = \begin{matrix} \\ j_1 \\ j_2 \\ j_n \end{matrix} \begin{matrix} \begin{matrix} k_1 & k_2 & k_n \end{matrix} \\ \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \end{matrix}$$

$$a^l = \begin{matrix} k_1 \\ k_2 \\ k_n \end{matrix} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad w^l_{jk} \cdot a^l_{kl}$$

$$b^l = \begin{matrix} j_1 \\ j_2 \\ j_n \end{matrix} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j \rightarrow z^l = w^l a^{l-1} + b^l$$

z = WEIGHTED INPUT

HADAMART PRODUCT: $s \odot t$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 \\ 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

BACKPROPAGATION FUNDAMENTAL EQUATIONS

BP1: ERROR IN THE OUTPUT LAYER

$$\delta^L_j = \frac{\partial C}{\partial a^L_j} \sigma'(z^L_j)$$

IF $C = \frac{1}{2} \sum_j (y_j - a^L_j)^2$

THEN $\partial C / \partial a^L_j = (a^L_j - y_j)$

$\delta^L = \nabla_a C \odot \sigma'(z^L)$ AS MATRIX FORM

WITH SIMPLIFIED COST:

$\delta^L = (a^L - y) \odot \sigma'(z^L)$

BP2: ERROR FROM ERROR IN THE NEXT LAYER (δ^{l+1})

$$\delta^l = \left((w^{l+1})^T \delta^{l+1}\right) \odot \sigma'(z^l)$$

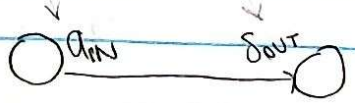
TRANSPOSED: MOVES THE ERROR BACKWARDS THROUGH THE NETWORK

BP3: CHANGE OF COST WITH BIAS

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

BP4: CHANGE OF COST WITH WEIGHT

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$



BACKPROPAGATION ALGORITHM:

(1) INPUT x: SET the CORRESPONDING ACTIVATION a FOR the INPUT LAYER. a FEEDS NEXT LAYER AS a^{l-1}.

(2) FEEDFORWARD: FOR EACH LAYER $l = 2, 3, \ldots, L$, COMPUTE

$$z^l = w^l a^{l-1} + b^l \text{ and } a^l = \sigma(z^l)$$

(3) OUTPUT ERROR δ^L: COMPUTE VECTOR

$$\delta^L = \underbrace{\nabla_a C}_{a^L - y \text{ (BP1)}} \odot \sigma'(z^L)$$

(4) BACKPROPAGATE the ERROR: FOR EACH LAYER $l = L-1, L-2, \ldots, 2$, COMPUTE

$$\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot \sigma'(z^l)$$

(5) OUTPUT: GRADIENT COSTS

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \text{ and } \frac{\partial C}{\partial b^l_j} = \delta^l_j$$

4 27 2020

MACHINE LEARNING UDEMY

BIG DATA AND PYSPARK

```
from pyspark import SparkContext

sc = SparkContext()

%%writefile example.txt
first line
second line
third line
```

```
textFile = sc.textFile('example.txt')
```

- textFile: RDD OBJECT
- sc: SPARK CONTEXT that CONNECTS to A SPARK CLUSTER
- textFile: METHOD
- 'example.txt': text object to CREATE the RDD OBJECT

RDD: ACTIONS: RETURNS VALUES
TRANSFORMATIONS: RETURNS POINTERS TO NEW RDD's

```
textfile.count() → 3 (LINES)
textfile.first() → 'first line'

secfind = textfile.filter(lambda
            line: 'second' in line)
secfind.collect() → ['second line']
secfind.count() → 1
```

RDD: RESILIENT DISTRIBUTED DATASET

TRANSFORMATION: SPARK OPERATION THAT PRODUCES AN RDD

ACTION: SPARK OPERATION THAT PRODUCES A LOCAL OBJECT

SPARK JOB: SEQUENCE of TRANSFORMATIONS WITH A FINAL ACTION

COMMON RDD TRANSFORMATIONS:

- filter()
- map()
- flatMap()
- sample()
- union()
- distinct()
- sortBy()
- reduceByKey

COMMON RDD ACTIONS:

- collect()
- take()
- top()
- takeSample()
- sum()
- mean()
- stdev()